Subject: Mode function for floating point arrays
Posted by Matthew Argall on Fri, 05 Jul 2013 18:55:23 GMT
View Forum Message <> Reply to Message

PEAMBLE:
I need a function that finds the mode of a floating point array. I have read David Fanning's article about integer arrays

http://www.idlcoyote.com/code_tips/mode.html

From this article about majority voting, it seems like "Hist_ND" works for floating point values, but I have no experience with the magic of HISTOGRAM

https://groups.google.com/forum/#!searchin/comp.lang.idl-pvw
ave/Mode$20of$20a$20floating$20point$20array/comp.lang.idl-p
vwave/YZK2ey-O5sE/9fLvx_AG2IAJ

QUESTION:
Here is my attempt. Can anyone make it better/faster?

```
;-------------------------------------------------
function mrmode, array, $
EPSILON=epsilon
   compile_opt idl2

   ;Number of points in ARRAY
   npts = n_elements(array)

   ;Default value for EPSILON
   if n_elements(epsilon) eq 0 then epsilon = 1d-5

   ;[index, count] for keeping track of mode statistics
   mode_count = lonarr(2, npts)

   ;Store first ~unique number. Count the how many ~unique numbers there are.
   mode_count[*,0] = [0,1]
   nunique = 1

   ;Step through all points in ARRAY
   for i = 1, npts - 1 do begin
      match_found = 0

      ;Try to pair the new point with other mode candidates
      for j = 0, nunique - 1 do begin
         if array[i] gt array[mode_count[0,j]]-epsilon && $
            array[i] lt array[mode_count[0,j]]+epsilon $
         then begin
```

```
            mode_count[1,j] += 1
            match_found = 1
         endif
      endfor

      ;If no match was found, create a new mode candidate
      if match_found eq 0 then begin
         mode_count[*,nunique] = [i,1]
         nunique += 1
      endif
   endfor

   ;Get the mode
   void = max(mode_count[1,*], iMode)
   mode = array[mode_count[0,iMode]]

   return, mode
end


 ;------------------------------------------------------------ ----------
;Example Program (IDL> .r mrmode) ////////////////////////////////////////
 ;------------------------------------------------------------ ----------
array = [1.2, 0.1, 3.3, 0.1, 2.0, 3.3, 4.8, 1.2, 0.1, 0.1, 6.7, 3.3]
mode = MrMode(array)
print, FORMAT='(%"The mode is: %f")', mode

end
```

Subject: Re: Mode function for floating point arrays
Posted by Rob Klooster on Mon, 08 Jul 2013 08:01:16 GMT
View Forum Message <> Reply to Message

Hi Matthew,

Histogram also works on floating point arrays, you just need to set the binsize:

```
hist = histogram(array, binsize=epsilon, locations=locations)
mode = locations[where(hist eq max(hist))]
```

Note that for small values of epsilon, the resulting histogram array can become very large.

Regards,
Rob.

Op vrijdag 5 juli 2013 20:55:23 UTC+2 schreef Matthew Argall het volgende:
> PEAMBLE:
>
> I need a function that finds the mode of a floating point array. I have read David Fanning's article about integer arrays
>
>
>
> http://www.idlcoyote.com/code_tips/mode.html
>
>
>
> From this article about majority voting, it seems like "Hist_ND" works for floating point values, but I have no experience with the magic of HISTOGRAM
>
>
>
> https://groups.google.com/forum/#!searchin/comp.lang.idl-pvw
ave/Mode$20of$20a$20floating$20point$20array/comp.lang.idl-p
vwave/YZK2ey-O5sE/9fLvx_AG2IAJ
>
>
>
>
>
> QUESTION:
>
> Here is my attempt. Can anyone make it better/faster?
>
>
>
> ;---
>
> function mrmode, array, $
>
> EPSILON=epsilon
>
> compile_opt idl2
>
>
>
> ;Number of points in ARRAY
>
> npts = n_elements(array)
>
>
>
> ;Default value for EPSILON

```
>
>     if n_elements(epsilon) eq 0 then epsilon = 1d-5
>
>
>
>     ;[index, count] for keeping track of mode statistics
>
>     mode_count = lonarr(2, npts)
>
>
>
>     ;Store first ~unique number. Count the how many ~unique numbers there are.
>
>     mode_count[*,0] = [0,1]
>
>     nunique = 1
>
>
>
>     ;Step through all points in ARRAY
>
>     for i = 1, npts - 1 do begin
>
>        match_found = 0
>
>
>
>        ;Try to pair the new point with other mode candidates
>
>        for j = 0, nunique - 1 do begin
>
>           if array[i] gt array[mode_count[0,j]]-epsilon && $
>
>              array[i] lt array[mode_count[0,j]]+epsilon $
>
>           then begin
>
>
>
>              mode_count[1,j] += 1
>
>              match_found = 1
>
>           endif
>
>        endfor
>
>
```

```
>
>         ;If no match was found, create a new mode candidate
>
>         if match_found eq 0 then begin
>
>             mode_count[*,nunique] = [i,1]
>
>             nunique += 1
>
>         endif
>
>     endfor
>
>
>
>     ;Get the mode
>
>     void = max(mode_count[1,*], iMode)
>
>     mode = array[mode_count[0,iMode]]
>
>
>
>     return, mode
>
> end
>
>
>
>
>
>  ;------------------------------------------------------------ ----------
>
> ;Example Program (IDL> .r mrmode) ////////////////////////////////////////
>
>  ;------------------------------------------------------------ ----------
>
> array = [1.2, 0.1, 3.3, 0.1, 2.0, 3.3, 4.8, 1.2, 0.1, 0.1, 6.7, 3.3]
>
> mode = MrMode(array)
>
> print, FORMAT='(%"The mode is: %f")', mode
>
>
>
> end
```

Subject: Re: Mode function for floating point arrays
Posted by Rob Klooster on Mon, 08 Jul 2013 08:24:08 GMT
View Forum Message <> Reply to Message

On second thought, it will be more efficient to treat the array as a sparse array and use value_locate, as in David's article:

http://www.idlcoyote.com/code_tips/valuelocate.html

```
sortedarray = array[Sort(array)]
arrayenum = sortedarray[Uniq(sortedarray)]
mappedarray = Value_Locate(arrayenum, array)
hist = histogram(mappedarray, min=0)
mode = arrayenum[where(hist eq max(hist))]
```

Maybe you can update the uniq function to accept a value for epsilon to decide whether two floating values are equal or not.

Regards,
Rob.

> Op maandag 8 juli 2013 10:01:16 UTC+2 schreef Rob Klooster het volgende:
> Hi Matthew,
>
>
>
> Histogram also works on floating point arrays, you just need to set the binsize:
>
>
>
> hist = histogram(array, binsize=epsilon, locations=locations)
>
> mode = locations[where(hist eq max(hist))]
>
>
>
> Note that for small values of epsilon, the resulting histogram array can become very large.
>
>
>
> Regards,
>
> Rob.
>
>
>
>

>
> Op vrijdag 5 juli 2013 20:55:23 UTC+2 schreef Matthew Argall het volgende:
>
>> PEAMBLE:
>
>>
>
>> I need a function that finds the mode of a floating point array. I have read David Fanning's article about integer arrays
>
>>
>
>>
>
>>
>
>> http://www.idlcoyote.com/code_tips/mode.html
>
>>
>
>>
>
>>
>
>> From this article about majority voting, it seems like "Hist_ND" works for floating point values, but I have no experience with the magic of HISTOGRAM
>
>>
>
>>
>
>>
>
>> https://groups.google.com/forum/#!searchin/comp.lang.idl-pvw
ave/Mode$20of$20a$20floating$20point$20array/comp.lang.idl-p
vwave/YZK2ey-O5sE/9fLvx_AG2IAJ
>
>>
>
>>
>
>>
>
>>
>
>>
>
>> QUESTION:

>
>>
>
>> Here is my attempt. Can anyone make it better/faster?
>
>>
>
>>
>
>>
>
>> ;---
>
>>
>
>> function mrmode, array, $
>
>>
>
>> EPSILON=epsilon
>
>>
>
>>   compile_opt idl2
>
>>
>
>>
>
>>
>
>>   ;Number of points in ARRAY
>
>>
>
>>   npts = n_elements(array)
>
>>
>
>>
>
>>
>
>>   ;Default value for EPSILON
>
>>
>
>>   if n_elements(epsilon) eq 0 then epsilon = 1d-5

```
>
>>
>
>>
>
>>
>
>>     ;[index, count] for keeping track of mode statistics
>
>>
>
>>     mode_count = lonarr(2, npts)
>
>>
>
>>
>
>>
>
>>     ;Store first ~unique number. Count the how many ~unique numbers there are.
>
>>
>
>>     mode_count[*,0] = [0,1]
>
>>
>
>>     nunique = 1
>
>>
>
>>
>
>>
>
>>     ;Step through all points in ARRAY
>
>>
>
>>     for i = 1, npts - 1 do begin
>
>>
>
>>        match_found = 0
>
>>
>
>>
>
>>
```

```
>
>>
>
>>        ;Try to pair the new point with other mode candidates
>
>>
>
>>        for j = 0, nunique - 1 do begin
>
>>
>
>>           if array[i] gt array[mode_count[0,j]]-epsilon && $
>
>>
>
>>              array[i] lt array[mode_count[0,j]]+epsilon $
>
>>
>
>>           then begin
>
>>
>
>>
>
>>
>
>>              mode_count[1,j] += 1
>
>>
>
>>              match_found = 1
>
>>
>
>>           endif
>
>>
>
>>        endfor
>
>>
>
>>
>
>>
>
>>        ;If no match was found, create a new mode candidate
```

```
>
>>
>
>>        if match_found eq 0 then begin
>
>>
>
>>            mode_count[*,nunique] = [i,1]
>
>>
>
>>            nunique += 1
>
>>
>
>>        endif
>
>>
>
>>    endfor
>
>>
>
>>
>
>>
>
>>    ;Get the mode
>
>>
>
>>    void = max(mode_count[1,*], iMode)
>
>>
>
>>    mode = array[mode_count[0,iMode]]
>
>>
>
>>
>
>>
>
>>    return, mode
>
>>
>
>> end
```

```
>
>>
>
>>
>
>>
>
>>
>
>>
>
>>   ;------------------------------------------------------- ----------
>
>>
>
>>  ;Example Program (IDL> .r mrmode) ////////////////////////////////////////
>
>>
>
>>   ;------------------------------------------------------- ----------
>
>>
>
>>  array = [1.2, 0.1, 3.3, 0.1, 2.0, 3.3, 4.8, 1.2, 0.1, 0.1, 6.7, 3.3]
>
>>
>
>>  mode = MrMode(array)
>
>>
>
>>  print, FORMAT='(%"The mode is: %f")', mode
>
>>
>
>>
>
>>
>
>>  end
```

Subject: Re: Mode function for floating point arrays
Posted by Matthew Argall on Tue, 09 Jul 2013 13:36:46 GMT
View Forum Message <> Reply to Message

It seems like VALUE_LOCATE and HISTOGRAM solutions would have large limitations. The bin size for HISTOGRAM would have to be "2*epsilon", which would rule out data with a large

dynamic range. Also, the bin should be centered on the data point so that two points falling within "epsilon" of one another do not get separated because the bins are offset.

For the sparse array idea, I would need to know beforehand what the is going to look like in order to create an acceptable map.

> Maybe you can update the uniq function to accept a value for epsilon to decide whether two floating values are equal or not.

This seems promising. I will check out the source code!

Subject: Re: Mode function for floating point arrays
Posted by David Fanning on Tue, 09 Jul 2013 14:01:28 GMT
View Forum Message <> Reply to Message

Matthew Argall writes:

>> Maybe you can update the uniq function to accept a value for epsilon to decide whether two floating values are equal or not.
>
> This seems promising. I will check out the source code!

FLOATS_EQUAL in the Coyote Library may be a place to start.

http://www.idlcoyote.com/programs/floats_equal.pro

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: Mode function for floating point arrays
Posted by Rob Klooster on Wed, 17 Jul 2013 14:08:37 GMT
View Forum Message <> Reply to Message

Op dinsdag 9 juli 2013 15:36:46 UTC+2 schreef Matthew Argall het volgende:
> It seems like VALUE_LOCATE and HISTOGRAM solutions would have large limitations. The bin size for HISTOGRAM would have to be "2*epsilon", which would rule out data with a large

dynamic range. Also, the bin should be centered on the data point so that two points falling within "epsilon" of one another do not get separated because the bins are offset.

The case of a large dynamical range is precisely the reason why I used VALUE_LOCATE instead of a plain HISTOGRAM with binsize set. Define the function like this:

```
function mode, array
  sortedarray = array[Sort(array)]
  arrayenum = sortedarray[Uniq(sortedarray)]
  mappedarray = Value_Locate(arrayenum, array)
  hist = histogram(mappedarray, min=0)
  return, arrayenum[where(hist eq max(hist))]
end
```

Example:

```
print, mode([1., 10.^8, 10.^8])
 1.00000e+008
print, mode([10.^8, 10.^8+1, 1.])
 1.00000e+008
print, mode([10.^8, 10.^8+10, 1.])
      1.00000 1.00000e+008 1.00000e+008
```

So in this case the machine precision is about 7 significant digits, as expected for floats. Note that two floats are only assumed equal when they have the exact same binary value.

Subject: Re: Mode function for floating point arrays
Posted by Matthew Argall on Fri, 19 Jul 2013 23:50:08 GMT
View Forum Message <> Reply to Message

It seems like the "goodness" of this lies in how well the UNIQ function can determine if two numbers are truly unique. Then, after that, how well Value_Locate can match unique values to their duplicates. Is that right?

> Note that two floats are only assumed equal when they have the exact same binary value.

I think there is more information in this sentence than I can grasp at the moment... Is there any reason to suspect that the precision of the result is less than the precision of the numeric type of the input array?

Subject: Re: Mode function for floating point arrays
Posted by Rob Klooster on Mon, 22 Jul 2013 11:24:49 GMT
View Forum Message <> Reply to Message

Op zaterdag 20 juli 2013 01:50:08 UTC+2 schreef Matthew Argall het volgende:
> It seems like the "goodness" of this lies in how well the UNIQ function can determine if two

numbers are truly unique. Then, after that, how well Value_Locate can match unique values to their duplicates. Is that right?

Exactly, UNIQ() is used for comparing floats to see if they are equal or not. You could change some lines in that function from:
  indices = where(q ne shift(q,-1), count)
to:
  indices = where(abs(q - shift(q,-1)) gt eps, count)
for a fixed value of eps. Be careful with this kind of comparisons, as the value to take for eps is not well defined. Take a look at this article which explains all the pitfalls when comparing floating point numbers:

 http://www.cygnus-software.com/papers/comparingfloats/compar ingfloats.htm

Value_locate() will work whatever the input is, since it does not look at exact matches. It will just find the interval to which a specific number belongs. You just need to make sure that the UNIQ() function outputs the lowest number of a particular bin.

>> Note that two floats are only assumed equal when they have the exact same binary value.
>
>
>
> I think there is more information in this sentence than I can grasp at the moment... Is there any reason to suspect that the precision of the result is less than the precision of the numeric type of the input array?

Again, have a look at the article. It will make things a bit clearer.

Rob.
